Model-Based real-time evaluation of security patterns: A SCADA system case study

Anas Motii[1], Agnès Lanusse[1], Brahim Hamid[2], Jean-Michel Bruel[2]

[1] CEA, LIST, Laboratory of Model Driven Engineering for Embedded Systems,
P.C. 174, Gif-sur-Yvette, 91191, France
{anas.motii,agnes.lanusse}@cea.fr
[2] IRIT, University of Toulouse
118 Route de Narbonne, 31062 Toulouse Cedex 9, France
{brahim.hamid,bruel}@irit.fr

Abstract. Securing critical systems such as cyber physical systems (CPS) is an important feature especially when it comes to critical transmitted data. At the same time, the implementation of security counter-measures in such systems may impact other functional or non-functional concerns. In this context, we propose a model-based approach for securing critical systems at early design stage. This approach combines security analysis and mitigation solution proposals with multi-concern architectural evaluation. It exploits two views of security counter-measures patterns: abstract and concrete. The abstract view is used to select relevant solutions to security requirements on a logical point of view. Then, the concrete view helps the architect evaluating different possible implementation alternatives against other design constraints. The modeling is based on accepted OMG standards such as UML and MARTE. In this paper, the approach is illustrated on a SCADA (Supervisory Control and Data Acquisition) system case study and a tool chain based on Papyrus UML supports the approach.

Keywords: Architecture evaluation, MBE for Cyber-Physical Systems, Model-Based System Analysis, Security Patterns, Model-Based Security Analysis

1 Introduction

Cyber-physical systems (CPS) consist of computational units controlling physical entities. The complexity of such systems during their design comes from the involvement of transdisciplinary concerns. Indeed, such systems must satisfy a number of requirements (real-time, physical, energy efficiency and others). In addition, critical cyber-physical systems have to satisfy assurance requirements (IEC 61508 and ISO 27005 [1], for dependability and security concerns). This brings the complexity of such systems to a higher level. In particular, security concerns have an impact on other concerns such as real-time performance. For example, encryption adds a delay to the transmission time of data from one point to the other and affects real-time constraints. Therefore, architects must apply trade-offs to satisfy functional requirements (real-time), and security requirements as two categories of constraints.

Model-Based System Engineering (MBSE) provides a useful contribution for the design and evaluation of secure systems. It makes easier the enactment of the separation of concern paradigm (security, real-time, performance, etc.). It helps the architect specify in a separate view non-functional requirements such as security at a high level of abstraction. Moreover, expertise and knowledge in system architecture and security can be captured within patterns that provide generic solutions for recurring problems. In particular for security, where protecting data and services is an important issue, security pattern catalogues [2] provide guidelines to build secure architectures.

Previous work have focused on security and real-time requirements separately: dependability and security modeling and analysis [3][4]; and real time requirements [5]. A survey of dependability modeling and analysis frameworks with UML can be found in [3]. It focuses on software systems Reliability, Availability, Maintenance and Safety (RAMS). In [4], the authors have extended MARTE with a Dependability Analysis and Modeling (DAM) UML profile and applied it to an intrusion-tolerant message service case study. In [5], the authors presented a staged approach to optimize the deployment in the context of real-time distributed systems.

Other works focused on large scale architecture optimization, decision and trade-off analysis [6][7]. In the automotive domain, a multi-objective automatic optimization approach based on EAST-ADL modeling is proposed [6]. It supports the evaluation of alternative architectures according to dependability, timing performance, cost etc. More specifically in security and performance interplay, the study in [7] focused on the analysis of the performance effects of security solutions modeled as UML non-functional aspects. It used SPT UML profile for annotating a UML design with schedulability, time and performance data. The resulting model and the security aspects were transformed separately and composed into one model which was then analyzed.

In this paper we present an approach to select and evaluate possible candidate improvements in order to find the best set of security patterns respecting timing constraints. To this end, we propose a model-based approach for the development of secure critical systems based on architectural evaluation driven by security concerns. This work is part of a more general process devoted to incremental pattern-based modeling and safety and security analysis for correct by construction systems design. In previous works, we have proposed a model-based approach for guiding the selection of security patterns based on risk analysis and pattern classification [8]. In a recent paper in [9], we proposed an approach to support Security, Dependability and Resource Tradeoffs using Pattern-based Development and Model-driven Engineering. In this paper, we go one step further, we study the impact of implementation alternatives of these security solutions onto the system architecture. A special emphasis is paid to timing performance concerns using model-based real-time evaluations. In this context, the system architect starts from a functional architecture and an abstract platform. The artifacts are abstract at this stage of development but contain temporal information (e.g., computation cost, deadlines and period of event for each function). Once security requirements are specified (resulting from a security risk analysis), several security pattern solutions are proposed from a repository of patterns. The real-time evaluation helps the architect to select the best candidates that respect timing concerns (e.g., maximum utilization capacity in the platform).

The remaining sections are organized as follows: Section 2 describes the SCADA system case study and its security issues. Section 3 illustrates the real-time evaluation approach of security solutions on the case study and gives its steps and modeling principles. Section 4 discusses the obtained experimental results. Section 5 concludes the paper and discusses future work.

2 Case study: SCADA system

2.1 Description

SCADA systems are meant to control processes through local controllers, acquiring field data and returning it to a SCADA master computer system. **Fig. 1** shows a typical SCADA system architecture. It consists of a SCADA master, an operator workstation and a number of field devices connected by a communication infrastructure. Field devices can be Programmable Logic Controllers (PLC), Remote Terminal units (RTU), sensors and actuators.

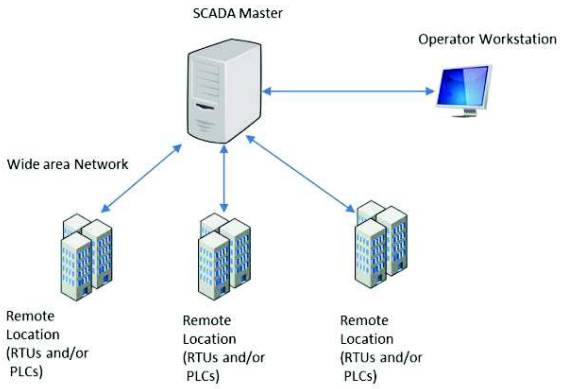


Fig. 1. A typical SCADA system architecture [10]

The SCADA master provides the operator with a Human-Machine Interface (HMI) through a work station to issue commands to PLCs and gather field data from them. PLCs are digital computers programmed to continuously monitor sensors and control actuators (e.g., valves, pumps, etc.). RTUs are used for converting sensor data into digital data. As SCADA systems cover large areas, they use Wide Area Networks (WAN). SCADA systems provide the following functionalities: data acquisition and handling (e.g., polling data from controllers, alarm handling, calculations, logging and archiving) on a set of parameters, typically those they are connected to.

2.2 SCADA security

It is very important for SCADA systems to be safe and reliable. They have a good reputation in this field. However, the key issue nowadays is SCADA security. Governments all over the world are worried about the security of SCADA systems that run

over critical infrastructures. First generation of SCADA systems were introduced in the 1970's and second generation in 1980's. Many of these are still in operation especially second generations. They relied on two approaches for security: (1) Security by isolation: based on the principle that if the system is not connected to the Ethernet then it cannot be attacked by external attackers. However it is still vulnerable to insider attacks. (2) Security through obscurity: based on the fact that SCADA systems used unusual programming languages and communication protocols. However this is also vulnerable to insider attackers who know about these technologies. In addition the documentation can be found on internet or can be stolen.
Third generation SCADA systems use standard IT technologies and protocols (e.g., organizational wireless networking, Microsoft windows, TCP/IP and web browsers as interfaces). The third generation systems which are web connected are integrated with an interface to second generation systems. Opening SCADA systems rises a major issue for guarantying security since the "security by isolation" principle is violated.
Proposing security solutions for critical systems, and in particular SCADA systems, requires an early architecture evaluation analyzing the impact of these solutions on quality attributes. In this paper, we treat one quality attribute which is real-time performance. The aim is to help the architect, at a high level design, selecting the best set of security solution implementations that respect timing requirements (if any). As mentioned earlier, the selection of security patterns is driven by a risk analysis performed in previous steps of the methodology [8]. This risk analysis follows a model-based implementation of EBIOS[1] methodology described in [11]. However this step is not described in this paper. The foundations of the approach are described in the next section.

3 Model-based Real-time evaluation of security pattern configurations

In this section we present the foundations of a model-based seamless approach for an incremental architecture securing process involving both: solutions identification, integration, evaluation and comparison. We present here the corresponding workflow, and illustrate each step of this process over the SCADA system case study.

3.1 Approach workflow overview

The process workflow proposed for architectural solutions evaluation follows 3 main steps (1 to 3) as illustrated in **Fig. 2**. Actually, the workflow itself is part of a more global process not described here that encompasses security analysis of design architecture proposal, and issues security requirements. Step0 here refers to a preliminary stage aimed at selecting appropriate pattern solutions satisfying these requirements. The main objective of the workflow presented below is to support the real-time evaluation

[1] EBIOS: Expression of Needs and Identification of Security Objectives from ANSSI, the french agency for security of information systems (Agence nationale de la sécurité des systèmes d'information).

of various possible security pattern configurations to assess their soundness regarding temporal (and possibly resource) constraints.

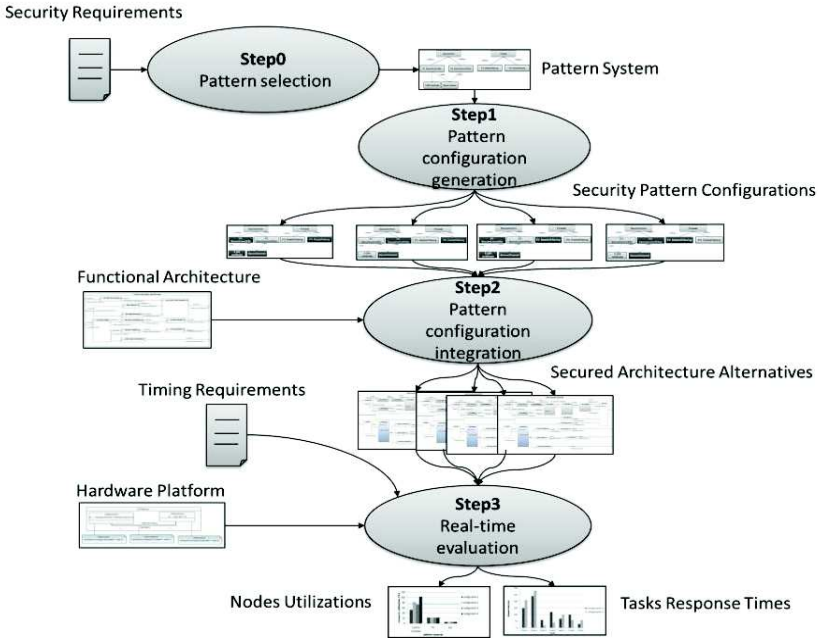


Fig. 2. Process of real-time evaluation of security pattern configurations

This seamless process relies on three main kinds of artifacts: (1) *functional architectures* to describe system and software functions, (2) *security patterns* to describe system security solutions and (3) *platform models* to describe hardware resources. The approach is centered on the concepts of security patterns. In this context, we consider the following definitions.

3.2 Definitions

Definition 1 (Security Pattern): a security pattern provides a generic solution of recurring security problem. Security patterns follow templates such as the ones proposed by GoF [12] and include several attributes e.g., "Name", "Context", "Problem", "Solution", "Consequences" and "See Also". "Structure" contains information about the functional structure of the pattern and uses generally semi-formal languages e.g., UML.to describe it. "Consequence" contains information about the impact of using this pattern on the target architecture quality attributes e.g., availability and performance.
The level of abstraction of the pattern depends on the detail of its solution. We distinguish: (1) abstract pattern providing an abstract solution without clear details of the used techniques (2) concrete pattern refining the solution provided by an abstract pattern possibly using other patterns. There are thus two types of relationships: refinement and usage relationships.

Definition 2 (System of security patterns): a System of security patterns is a collection of security patterns with their relationships. In our context, patterns have different refinement alternatives.
Definition 3 (System of security patterns configuration): A configuration is a subset of a System of security patterns. It will be used to specify the possible refinement alternatives (concrete patterns). It will be also referred to as "security solution alternative".
Definition 4 (Pattern integration): pattern integration means refining the functional architecture by adding security pattern functions. Each security solution alternative is integrated producing secured architecture alternatives. In the context of MDE, pattern integration is a "model refinement".

3.3 Process description

As stated earlier, the evaluation process is composed of three main steps (see Fig.2), and a preliminary one for patterns selection. This process considers as inputs security requirements resulting from prior risk analysis and the design model.
In the case of SCADA systems, such security requirements can be: (1) There should be a mechanism for secure communication that guarantees data integrity, confidentiality and authenticity, (2) There should be a mechanism that protects against denial of service attacks at the level of the SCADA master.

Step 0 (Pattern selection).
Selecting appropriate security solutions, here presented as security patterns, from security requirements is an important step during the development of secure software and systems. In this context, we use the selection method described in [8] which is based on the use of risk analysis to derive security properties and constraints; along with pattern selection principles using pattern classifications [13][14] to select concrete security patterns. The method is based on a library of patterns stored in the SEMCO repository [15]. The System and software Engineering Pattern Metamodel (SEPM) [15] is used to model Security and Dependability (S&D) patterns which are then stored in a repository. Patterns provide their functionalities through interfaces. Their characteristics are described by properties.
After analyzing security requirements, the architect identifies a set of security patterns along with their refinement alternatives, i.e. concrete patterns. It is important to note that the selection of security patterns takes into account conflicts due to inconsistencies between patterns. For example, Limited view and Full view pattern are conflictual by nature so that implementing both of them in a system will surely bring inconsistencies. The search in the repository leads to the identification of two abstract patterns:

- *SecureComm* pattern [2]: ensures that data passing across a secure network is secure. It can be refined by two patterns: *SecureCommSSL* (P1) and *SecureCommIPsec* (P2). *SecureCommSSL* uses *X.509 certificates* for authentication and *secure channel* for creating a cryptographic tunnel.
- *Firewall* pattern [2]: restricts access to internal networks which can be refined by *PacketFilter* (P3) and *StatefulFiltering* (P4).

The result of this step is the System of security patterns represented in **Fig. 3**.

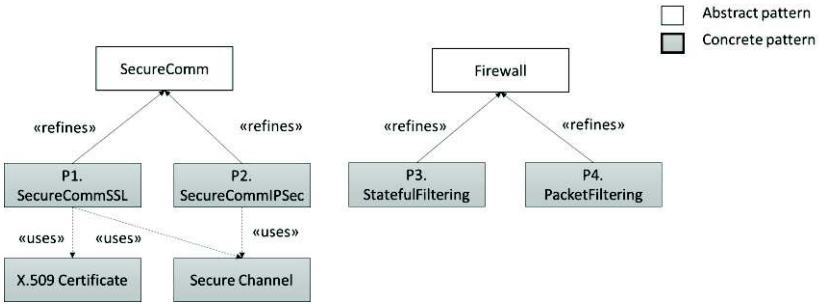


Fig. 3. System of security patterns

Step 1 (System of patterns configuration generation).
The goal of this step is to create the possible security solution alternatives from a system of patterns using system of patterns configuration management based on variability models.

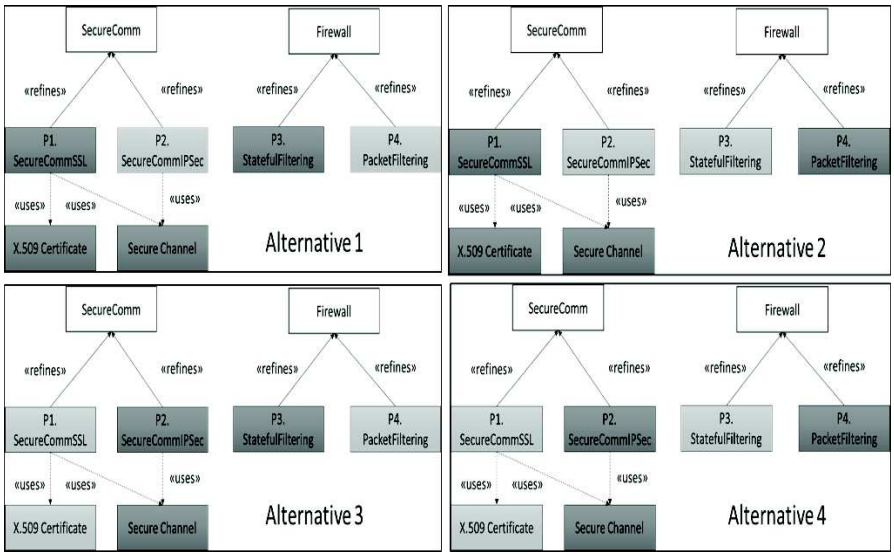


Fig. 4. Security solution alternatives generated from the pattern system

Security variability consists of documenting all alternative security solutions for a given problem. Security variability is important in the context of architecture design because it helps the architect understanding (1) which security solution varies and how it varies, (2) if there are security solutions satisfying several security objectives, (3) possible conflicts between security solutions, (4) security solutions that supports others.
Several works have treated this concern [16] [9]. In [16], the authors present a modeling approach based on aspect engineering. In [9], the author have presented an algorithm for pattern system configuration management. It takes as input a pattern system with its

relationships, a base configuration and a reference kind (i.e. relationship type); and outputs a security solution alternative.

In the context of the paper [9] is used. For the case study, **Fig. 3** can be considered as a possible security variability model. Then, **Fig. 4** shows the corresponding possible security solution alternatives. Each system of security solution alternative consists of a set of concrete patterns in dark grey.

Step 2 (Pattern configuration integration).

The goal of this step is to integrate each security solution alternative into the functional architecture thus obtaining refined design architecture candidates.

The difficulty of this step is not only the verification of the correct integration of the pattern but also the management of possible conflicts between the functional architecture and a security pattern or between security patterns themselves.

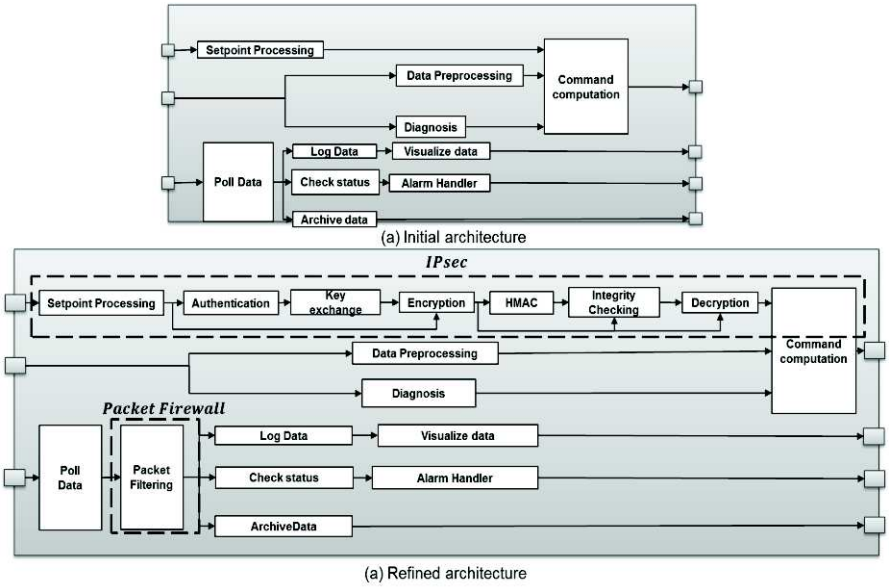


Fig. 5. SCADA functional architecture before (a) and after the integration of Alternative 4 (b)

The integration of concerns; and security in particular has been treated in [17][18][19]. Pattern integration consists of composing security patterns with a functional architecture. It requires techniques such as: role bindings merge techniques, checking techniques prior and after the integration to detect possible conflicts.

In the context of this work, [19] is used. Each security solution alternative is applied into the SCADA system functional architecture. **Fig. 5** depicts the initial SCADA functional architecture (a) and the result of the integration (b) of *security solution alternative 4*.

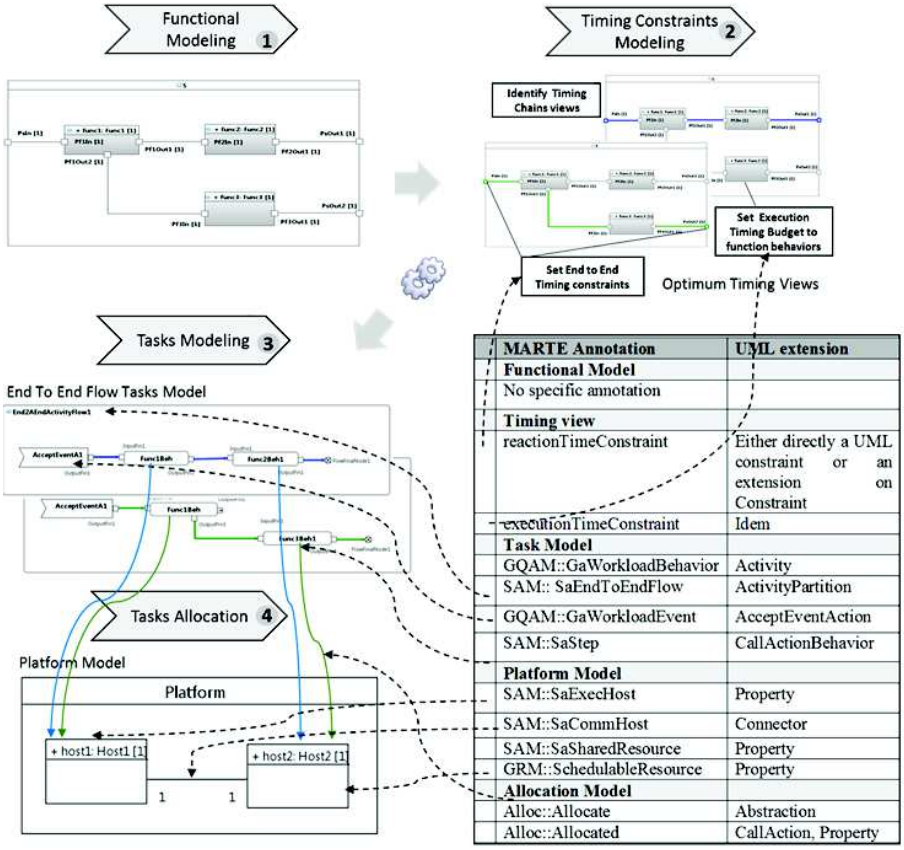
Step 3 (Real-time evaluation).
Real-time evaluation principles are based on works by [20] with the Optimum methodology[2]. This methodology was previously applied in other modeling contexts such as EAST-ADL in the MAENAD project. In this approach design models are evaluated at early stages to help make architectural decisions. Here we use it on UML design models and patterns stored in SEMCO Library.

Modeling principles.
The Optimum methodology is used to build a task model from the design in order to evaluate, compare architectural solutions and/or optimize deployment. We describe here the modeling principles and main steps of the methodology. Note that in our context, it is applied to high level functional design to get preliminary decisions on the overall architecture security improvement solutions. The input is thus a functional view of the application and patterns annotated using MARTE profile[3].
A task model is obtained following four steps: (1) identification of event-chains in the functional model, (2) specification of timing constraints (on event-chains and on behaviors corresponding to functions), (3) computation of a MARTE task model (end-to-end flows), and (4) tasks Allocation Specification (tasks on nodes).
To support the approach several diagrams are used: composite diagrams for 1) and 2); activity diagrams for 3); and composite diagrams showing task model and platform model together with allocation links for 4). This Optimum workflow and MARTE notations used are summarized in **Fig. 6**.
Event chains identification. The functional organization of the application is described in a Composite diagram showing functions and their connections. From this global view several timing views corresponding to end-to-end flows are selected.
Timing constraints setup. Selected event chains are then tagged to setup timing constraints. MARTE annotations are added to these diagrams to set: (1) event chains timing constraints (between 2 ports), (2) execution time constraints on functions (actually expected for the behavior implementing the function).
Task model setup. The task model structure is described using activity diagrams and can be directly obtained from the event chains specifications above. Each of them is translated into a MARTE end-to-end event flow. Each flow is activated by the reception of an event and described by the consequent behaviors implementing the various functions traversal connected through connectors.
MARTE annotations are used to: (1) characterize a timing configuration, (2) specify a data arrival pattern for the activating event (workflowEvent) and (3) specify constraints on the different steps (behaviors involved in the event flow).

---

[2] Optimum methodology is developed at LIST CEATech and is integrated within Papyrus open-source modeling tool.

[3] MARTE profile is a standard from the OMG (UML Profile for MARTE™: Modeling and Analysis of Real-time Embedded Systems™). http://www.omgmarte.org/



MARTE Annotation	UML extension
Functional Model	
No specific annotation	
Timing view	
reactionTimeConstraint	Either directly a UML constraint or an extension on Constraint
executionTimeConstraint	Idem
Task Model	
GQAM::GaWorkloadBehavior	Activity
SAM:: SaEndToEndFlow	ActivityPartition
GQAM::GaWorkloadEvent	AcceptEventAction
SAM::SaStep	CallActionBehavior
Platform Model	
SAM::SaExecHost	Property
SAM::SaCommHost	Connector
SAM::SaSharedResource	Property
GRM::SchedulableResource	Property
Allocation Model	
Alloc::Allocate	Abstraction
Alloc::Allocated	CallAction, Property

Fig. 6. Using MARTE to set timing constraints

Allocation model setup. Finally an allocation model is described in a composite diagram that shows the allocation between functions (actually the tasks corresponding to their behaviors) onto a platform model.
MARTE annotations are used to: (1) set allocation relations and (2) set hardware architecture characteristics on execution hosts and communication channels.

Real-time evaluation steps.
The modeling principles described earlier allow the specification of task and allocation models exploitable by scheduling algorithms such as Rate Monotonic scheduling (RMS) and offset-based scheduling [21][22]. Let P be a platform consisting of connected nodes $\{N_1, N_2, \dots, N_a\}$ and F be a set of functions $\{f_1, f_2, \dots, f_b\}$. Each function has a computation cost c_{f_i}. The execution nodes are connected through buses. Both nodes and buses have a maximal capacity that must not be exceeded. In the context of multiple processors, each node runs an independent real-time operating system. Each platform node N_i executes a set of tasks $T_i = \{t_{i,1}, t_{i,2}, \dots, t_{i,ni}\}$. Each task consists of a subset of functions from F. Each task $t_{i,j}$ has an activation period $P_{i,j}$, an execution

time $c_{i,j}$ (computed as the sum of computation costs of all allocated function) and a deadline$D_{i,j}$. Real-time evaluation consists of two steps:

1. Preliminary evaluation. A preliminary real-time evaluation is used to compute nodes utilizations. If one of the refined architectures exceeds the node maximum capacity, it is rejected. The node utilization is computed as: $U_{Ni} = \sum_{t_{i,j} \in Ti} \frac{c_{i,j}}{P_{i,j}}$

In case of RMS, the utilization U_{Ni} for each node must at least be: $U_{Ni} \leq n(2^{\frac{1}{n}} - 1)$ with n being the number of tasks assigned to Ni. If not, the node is overloaded and *response time analysis* is not performed.

2. Response time analysis. The response time analysis is performed for architectures succeeding the preliminary evaluation. At this step, response time analysis is performed following the principles proposed in [22] for distributed systems. It concerns the computation of the worst case response time $R_{t_{i,j}}$ of every task. All response times must verify: $R_{t_{i,j}} \leq D_{i,j}$. If not, the tasks are not schedulable and the corresponding architecture is rejected.

In the context of this work, we use RMS for preliminary evaluation and offset-based scheduling for response time analysis using QOMPASS tool that supports Optimum methodology.

4 Preliminary experimental results

As a preliminary experiment, we apply the approach to a SCADA system case study. **Fig. 7** shows the input functional architecture together with hardware platform. The functional model contains ten functions in three transactions with their deadlines and trigger periods. The hardware topology in the platform contains a SCADA master and a PLC connected with Modbus. The partitioning of functions into tasks and assignment of tasks onto hosts is also shown. In addition, the signal between "Set point processing" and "Command computation" is mapped onto a message. The execution budgets of the functions, the assigned tasks and hosts are showed in **Table 1**. The values of the SCADA function timing parameters are based on IEEE 1646 standard [23] specifying communication deadlines and IEC 61850 [24] specifying communication network delays in different information categories.

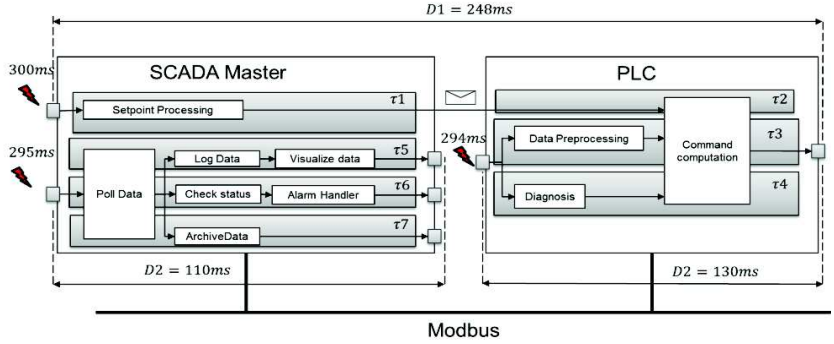


Fig. 7. Input functional architecture, hardware platform and deployment

Similarly, the timing and placement parameters of the used security patterns are showed in **Table 2**. The concrete patterns have the same functions but have different execution times (two execution time columns). The timing parameters are based on a review of technical reports of SSL/IPsec [25], and stateful/packet firewall [26].
One important point is that the experiment has required some effort in quantifying real-time parameters of security pattern functions. Some functions execution times were estimations and averages. For example, in *SecureComm* pattern function "HMAC" does not have the same execution time as it depends on the used algorithm (e.g., HMAC-SHA-1-96, HMAC-MD5). However, we believe that estimations and averaging is enough as the approach is meant for high level evaluation and architecture decision making. For example, if none of the security solution alternatives respected the timing requirements because of overload; the architecture of SCADA can be rethought leading to adding an execution node.

Table 1. Timing parameters and deployment of SCADA functions

Functions	Execution time	Task	Host
Setpoint Processing	8.7	$\tau 1$	*SCADA master*
Poll Data	9.6	$\tau 5, \tau 6, \tau 7$	*SCADA master*
Log Data	8.5	$\tau 5$	*SCADA master*
Check Status	9.6	$\tau 6$	*SCADA master*
Visualize Data	10.5	$\tau 5$	*SCADA master*
Alarm Handler	10.3	$\tau 6$	*SCADA master*
Archive Data	9.5	$\tau 7$	*SCADA master*
Command Computation	10	$\tau 2, \tau 3, \tau 4$	*PLC*
Data Preprocessing	9.5	$\tau 3$	*PLC*
Diagnosis	8.9	$\tau 4$	*PLC*

Table 2. Timing parameters and deployment of security pattern functions

Patterns	Functions	Execution times		Task
		(1)	(2)	
SecureCommSSL (1) SecureCommIPsec (2)	Authentication	9.7	38.7	$\tau 1$
	Key exchange	10.1	39.6	$\tau 1$
	Encryption	9.9	9.9	$\tau 1$
	HMAC	9.2	9.2	$\tau 1$
	Decryption	10.3	10.3	$\tau 2$
	Integrity checking	10.2	10.2	$\tau 2$
PacketFiltering (1) StatefulFiltering (2)	Filtering	10	40	$\tau 7$

4.1 Results

The preliminary analysis consists in evaluating the placement of SCADA and pattern functions on hosts described in **Table 1** for each security solution alternative (1, 2, 3

and 4) in **Fig. 4**. The left side of **Fig. 8** shows the node utilization results of each alternative. The utilization bound of the SCADA master and PLC are up to 75.68% (four tasks) and 77.97% (three tasks) respectively. Security solution alternatives 2 and 4 are rejected because the SCADA master utilization in the two cases (83.33% and 103.33%) exceeds the threshold. Response time analysis given in [22] is performed for security solution alternatives 1 and 3 since they pass the preliminary evaluation. Task $\tau 2$ response time is up to 280ms in alternative 3 and violates its deadline of 248ms. This is due to the offset added by task $\tau 2$ and the message transmission time. All tasks of configuration 1 respect their deadline: $\tau 1$ (150ms), $\tau 2$ (240ms), $\tau 3$ (60ms), $\tau 4$ (120ms), $\tau 5$ (70ms), $\tau 6$ (100ms) and $\tau 7$ (30ms). From the evaluations, alternative 1 is the best security solution alternative that fulfils security requirements and respects real-time constraints.

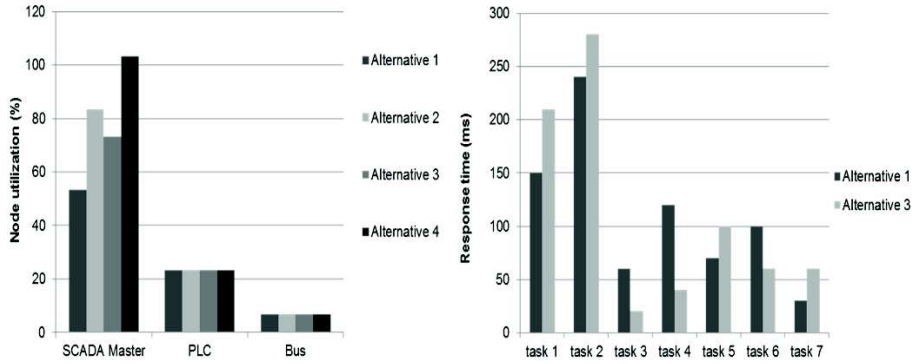


Fig. 8. Node utilization for security solution alternatives 1, 2, 3 and 4, and tasks response times for alternatives 1 and 3

4.2 Discussion

From this first experimentation, we conclude that the approach fulfils the objective of finding the best set of security patterns respecting timing constraints.

The work has two main contributions: (1) the proposal of abstract security pattern solutions fulfilling security requirements and (2) the evaluation of the possible implementations fulfilling real-time requirements by the integration of possible security solution alternatives. In this context, this work can be beneficial to resource constrained embedded systems e.g., automotive, avionics. For instance in EAST-ADL, trade-off analysis is performed for one design model with different parameters whose values determine whether the design satisfies the model or not. Our work adds a step forward which is the evaluation of different design alternative models against non-functional concerns (security in this paper). This work can benefit from EAST-ADL concepts for configurations management using features diagrams.

Acknowledgements. This work is conducted in the context of a Ph.D. thesis funded by CEA LIST and co-leaded by CEA (LISE) and IRIT (MACAO).

5 Conclusion and future work

The paper presents a model-based approach for evaluating security solutions based on patterns applied to a SCADA system case study. It shows the applicability of the approach. The main benefits are to provide a tooling support for early evaluation of different implementation of security measures using: pattern composition and integration, automatic configuration generation and evaluation. The evaluation focuses on temporal performance concerns. This work is part of a process devoted to incremental pattern-based modeling and safety and security analysis for correct by construction systems design. The results obtained help the designer select appropriate design solution to reinforce security. The methodology relies on UML/MARTE for modeling and makes extensive use of MARTE to perform architectural evaluation for timing concerns. This work will be extended to address other concerns (e.g., cost, reliability, memory consumption, power supply).

References

1. ISO/IEC 27005: Information technology — Security techniques — Information security risk management. (2011).
2. Fernandez, E.B.: Security Patterns in Practice: Designing Secure Architectures Using Software Patterns. Wiley Publishing (2013).
3. Bernardi, S., Merseguer, J., Petriu, D.C.: Dependability modeling and analysis of software systems specified with UML. ACM Comput Surv. 45, 2 (2012).
4. Bernardi, S., Merseguer, J., Petriu, D.C.: A dependability profile within MARTE. Softw. Syst. Model. 10, 313–336 (2011).
5. Mehiaoui, A., Wozniak, E., Piergiovanni, S.T., Mraidha, C., Natale, M.D., Zeng, H., Babau, J.-P., Lemarchand, L., Gérard, S.: A two-step optimization technique for functions placement, partitioning, and priority assignment in distributed systems. In: SIGPLAN/SIGBED Conference on Languages, Compilers and Tools for Embedded Systems 2013, LCTES '13, Seattle, WA, USA, June 20-21, 2013. pp. 121–132 (2013).
6. Walker, M., Reiser, M.-O., Tucci-Piergiovanni, S., Papadopoulos, Y., Lönn, H., Mraidha, C., Parker, D., Chen, D., Servat, D.: Automatic optimisation of system architectures using EAST-ADL. J. Syst. Softw. 86, 2467–2487 (2013).
7. Petriu, D.C., Woodside, C.M., Petriu, D.B., Xu, J., Israr, T., Georg, G., France, R., Bieman, J.M., Houmb, S.H., Jürjens, J.: Performance analysis of security aspects in UML models. Proc. 6th Int. Workshop Softw. Perform. 91–102 (2007).
8. Motii, A., Hamid, B., Lanusse, A., Bruel, J.-M.: Guiding the Selection of Security Patterns Based on Security Requirements and Pattern Classification. In: Proceedings of the 20th European Conference on Pattern Languages of Programs. p. 10:1–10:17. ACM, New York, NY, USA (2015).
9. Hamid, B.: Interplay of Security&Dependability and Resource Using Model-Driven and Pattern-Based Development. In: 2015 IEEE Trustcom/BigDataSE/ISPA. pp. 254–262 (2015).

10. Technical Information Bulletin 04-1: Supervisory Control and Data Acquisition (SCADA) System, (2004).
11. Abdallah, R., Motii, A., Yakymets, N., Lanusse, A.: Using Model Driven Engineering to Support Multi-paradigms Security Analysis. In: Desfray, P., Filipe, J., Hammoudi, S., and Pires, L.F. (eds.) Model-Driven Engineering and Software Development. pp. 278–292. Springer International Publishing (2015).
12. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-oriented Software. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1995).
13. Fernandez, E.B.: Using security patterns to develop secure systems. In: Software engineering for secure systems. Industrial and research perspectives. pp. 16–31 (2011).
14. Bunke, M., Koschke, R., Sohr, K.: Organizing security patterns related to security and pattern recognition requirements. Int. J. Adv. Secur. 5, (2012).
15. Hamid, B., Percebois, C.: A Modeling and Formal Approach for the Precise Specification of Security Patterns. In: Jürjens, J., Piessens, F., and Bielova, N. (eds.) Engineering Secure Software and Systems. pp. 95–112. Springer International Publishing (2014).
16. Dai, L.: Security Variability Design and Analysis in an Aspect Oriented Software Architecture. In: Third IEEE International Conference on Secure Software Integration and Reliability Improvement, 2009. SSIRI 2009. pp. 275–280 (2009).
17. Alam, O., Kienzle, J., Mussbacher, G.: Concern-Oriented Software Design. In: Moreira, A., Schätz, B., Gray, J., Vallecillo, A., and Clarke, P. (eds.) Model-Driven Engineering Languages and Systems. pp. 604–621. Springer Berlin Heidelberg (2013).
18. Nguyen, P.H., Yskout, K., Heyman, T., Klein, J., Scandariato, R., Le Traon, Y.: Model-Driven Security based on A Unified System of Security Design Patterns. (2015).
19. Hamid, B., Percebois, C., Gouteux, D.: A Methodology for Integration of Patterns with Validation Purpose. In: Proceedings of the 17th European Conference on Pattern Languages of Programs. p. 8:1–8:14. ACM, New York, NY, USA (2012).
20. Mraidha, C., Tucci-Piergiovanni, S., Gerard, S.: Optimum: A MARTE-based Methodology for Schedulability Analysis at Early Design Stages. SIGSOFT Softw Eng Notes. 36, 1–8 (2011).
21. Harbour, M.G., Gutiérrez, J.J., Drake, J.M., Martínez, P.L., Palencia, J.C.: Modeling distributed real-time systems with MAST 2. J. Syst. Archit. 59, 331–340 (2013).
22. Tindell, K., Clark, J.: Holistic Schedulability Analysis for Distributed Hard Real-time Systems. Microprocess Microprogram. 40, 117–134 (1994).
23. IEEE Standard Communication Delivery Time Performance Requirements for Electric Power Substation Automation. IEEE Std 1646-2004. 0_1-24 (2005).
24. CODE, P.: Communication networks and systems in substations–Part 5: Communication requirements for functions and device models. (2003).
25. Alshamsi, A., Saito, T.: A technical comparison of IPSec and SSL. In: 19th International Conference on Advanced Information Networking and Applications (AINA'05) Volume 1 (AINA papers). pp. 395–398 vol.2 (2005).

26. Design and Performance of the OpenBSD Stateful Packet Filter (pf), http://www.benzedrine.ch/pf-paper.html.